This Page is Inserted by IFW Indexing and Scanning Operations and is not part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☐ FADED TEXT OR DRAWING
- ☐ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☐ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☐ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☐ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: _____

IMAGES ARE BEST AVAILABLE COPY.
As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.

L Number	Hits	Search Text	DB	Time stamp
4	55	wireless near9 (interworking adj function$5)	USPAT	2004/08/10 14:22
5	15	application$5 near9 (interworking adj function$5)	USPAT	2004/08/10 14:23
6	6	provision$6 near9 (interworking adj function$5)	USPAT	2004/08/10 14:50
7	8	provision$6 with (interworking adj function$5)	USPAT	2004/08/10 14:24
12	0	proxy near9 (interworking adj function$5)	USPAT	2004/08/10 14:50
13	0	proxy with (interworking adj function$5)	USPAT	2004/08/10 14:50
14	37	proxy and (interworking adj function$5)	USPAT	2004/08/10 14:51
15	43	(gateway) with (interworking adj function$5)	USPAT	2004/08/10 14:51
17	31	single adj point adj interface	USPAT	2004/08/10 15:07
18	85	(msc (mobile adj switch adj center$3)) with provision$5	USPAT	2004/08/10 15:09
19	0	internet near3 feature near3 provid$5 near3 web	USPAT	2004/08/10 15:10
20	12	internet near9 feature near9 provid$5 near9 web	USPAT	2004/08/10 15:10
21	0	internet near9 inheren$6 near9 provid$5 near9 web	USPAT	2004/08/10 15:10
22	9	internet near9 inheren$6 near9 web	USPAT	2004/08/10 15:10
23	1	6615038.pn. and ((msc (mobile adj switch$5)) with (configu$9 upgrad$5))	USPAT	2004/08/10 15:26
24	0	provisioning near web near service$3	USPAT	2004/08/10 15:21
25	14	provisioning near web near service$3	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/08/10 15:26
26	0	(provisioning near web near service$3) same (customer$3 near9 service$5)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/08/10 15:26
30	8	(provision$5 near3 service$5) near4 (mobile adj switch$5)	USPAT	2004/08/10 15:50

Sponsored Links

Affordable HA Clustering
Cluster your existing Servers & Applications with RSF-1

Nucleus RTOS
Project and Consulting Services Embedded real-time DSP solutions

Beowulf Linux Clusters
Preconfigured HPC Beowulf clusters optimized for performance

Agilent Protocol Analyzer
Change the way you troubleshoot ATM and VoIP networks.

article insider
Real People... Sharing Real Knowledge

internet.com
Developer Downloads

provisioning

Last modified: Tuesday, May 04, 2004

(n.) *(1)* The process of providing users with access to data and technology resources. The term typically is used in reference to enterprise-level resource management. Provisioning can be thought of as a combination of the duties of the human resources and IT departments in an enterprise, where (1) users are given access to data repositories or granted authorization to systems, applications and databases based on a unique user identity, and (2) users are appropriated hardware resources, such as computers, mobile phones and pagers. The process implies that the access rights and privileges are monitored and tracked to ensure the security of an enterprise's resources.

(2) The process of providing customers or clients with accounts, the appropriate access to those accounts, all the rights associated with those accounts, and all of the resources necessary to manage the accounts. When used in reference to a client, provisioning can be thought of as a form of customer service.

•E-mail this definition to a colleague•

Sponsored listings

Dealtime: Microsoft Provisioning System For Pc Products - Compare prices with DealTime and find discount prices on computer, software, and office products.

Dealtime: Sun N1 Provisioning Srvr 3.0 Media Products - Compare prices with DealTime and find discount prices on computer, software, and office products.

For internet.com pages about ***provisioning*** ***CLICK HERE****. Also check out the following links!*

Related Categories

Business Computing
Digital Rights Management (DRM)
Security
Telecommunications

Related Terms

authorization
credential
CRM

International
Internet Lists
Internet News
Internet Resources
IT
Linux/Open Source
Small Business
Windows Technology
Wireless Internet
xSP Resources

Search internet.com
Advertise
Corporate Info
Newsletters
Tech Jobs
E-mail Offers

internet commerce
Be a Commerce Partner
Corporate Gifts
Cheap Web Hosting
Submit Your Site
Dedicated Servers
Franchise Directory
Custom Web Design
Find IT Products
Web directory
Calling Cards
eCommerce Web Hosting

LINKS = Great Page!

Inside ID Home Page
News and information for identity management specialists.

enterprise

identity

(Webopedia)
Give Us Your Feedback

Shopping
provisioning Products
Compare Products,Prices and Stores

Shop by Category:
Network Management Tools
196 Model Matches

Business and Productivity Software
81 Model Matches

Wi-FiHotSpotList.com™
The Definitive Source For Wi-Fi Access Points
Close Search
The Form below will allow you to searc
in the US. For a broader search click h
City i.e.New York

JupiterWeb networks:
internet.com | EARTHWEB | dev | ClickZ

Search JupiterWeb: Find

Jupitermedia Corporation has four divisions:
JupiterWeb, JupiterResearch, JupiterEvents and JupiterImages

Copyright 2004 Jupitermedia Corporation All Rights Reserved.
Legal Notices, Licensing, Reprints, & Permissions, Privacy Policy.

Jupitermedia Corporate Info | Newsletters | Tech Jobs | E-mail Offers

Microsoft.com Home | Site Map

msdn

MSDN Home | Developer Centers | Library | Downloads | Code Center | Subscriptions | MSD

Search for

MSDN Magazine Go

Advanced Search

MSDN® Magazine
The Microsoft Journal for Developers

MSDN Home > MSDN Magazine > December 2002

MSDN Magazine Home
December 2002
Search
Source Code
Back Issue Archive
Column Archive
Next Issue
Subscribe
Order Back Issues
Reader Services
Special CD Offer
Meet the Staff
Media Kit
Submit an Article
Write to Us
Corrections
MSJ Archive
MIND Archive
Magazine Newsgroup

Adver

100+
in 8

TEC
Octob
New
http://ww

PROVISIONING

Use Web Services Provisioning to Control Access, Usage, and Billing on Your Site

Chandu Thota

Print

This article assumes you're familiar with C#, SQL, and SOAP

Level of Difficulty 1 2 3

Download the code for this article: WebServicesProvisioning.exe (139KB)

SUMMARY Building Web Services to provide enterprise-level solutions is only the first step. You need to take care of the infrastructure aspects of your solution as well, including provisioning, billing, security, and reporting. In this article, the author uses the .NET Framework and SQL Server 2000 to design a provisioning system that will take care of all these housekeeping tasks. He discusses the general requirements of a Web Service provisioning system, walks through the implementation, and then outlines various scenarios for putting this system to work.

Like just about everyone else these days, I'm under the spell of Web Services. As a managed developer, I'm even more fascinated because the Microsoft® .NET Framework makes it simple to put Web Services to use. The promise of these technologies—enterprise application integration and software as a service—make them a viable business solution. Of course, the moment you start thinking about developing Web Services as a real-world business solution, you realize that you're going to have to take care of various real-world business aspects such as provisioning, billing, security, and reporting.

In this article, I will discuss how to design a provisioning system using the .NET Framework and SQL Server™ 2000. I will begin by discussing the requirements of a Web Service provisioning (WSP) system, which I'll refer to simply as WSP. Then I'll examine the design aspects and coding. Finally, I will outline various scenarios for putting this system to work.

A WSP Primer

WSP is the technology I created for use in a pay-per-use Web Services scenario. WSP can be explained as a utility or supporting architecture that enables Web Services provisioning and subscription services for the two important roles that exist in the world of Web Services: the role of provider and the role of consumer. Web Service providers are service sellers who own, operate, and manage the Web Services. Web Service consumers subscribe, consume, and pay for the Web Services. It is also possible for a user to be both a provider and a consumer.

From an architectural perspective, WSP consists of five major applications

working together. These include the service catalog, service subscription manager, service provision manager, metering and billing engine, and WSP runtime. Let's examine each of these applications briefly.

Service catalog This contains published services and the corresponding technical and licensing details. This is the first step for both service providers and service consumers in publishing and consuming a Web Service. A private or public Universal Description, Discovery, and Integration (UDDI) instance can also be used as a service catalog. Service providers usually publish the services and consumers discover them by browsing the catalog.

Service subscription manager This component automates the subscription and enrollment process. Service consumers discover the services and subscribe to them using the subscription manager. The subscription manager usually implements the licensing and contractual aspects of the WSP.

Provision manager Automation of the service publishing for the service providers is the role of this component. A provision manager is also used to specify the licensing and rating information.

Metering and billing engine The responsibility for processing the Web Service usage logs, calculating the rates based on the licensing agreements, and presenting the bills to the service providers and consumers on a periodic basis fall on this component.

WSP runtime This is the most important piece of the WSP system. It is responsible for intercepting the incoming SOAP messages for authorization, license evaluation, and logging the service usage.

Now, let's take some time to understand the WSP process flow. A service publisher logs on to the WSP system and uses provision manager to publish the service in the service catalog. This publication process includes specifying service details such as service description, service interface (the WSDL), service end-point, security model, licensing model, and access fees.

Once the publishing process has completed, the Web Service can only be accessed through the WSP runtime engine, with all incoming requests going through the WSP runtime. With each incoming request, the WSP runtime engine performs security (identity) and licensing checks before the request reaches the actual Web Service. All requests coming into the server must satisfy both the security and licensing checks; otherwise, access to the Web Service will be denied.

When an incoming request satisfies both security and licensing checks, the request is fulfilled and the usage details, such as the consumer identity and usage time, are logged into the database using the logging application.

The publisher can access the Web Service usage logs using the logging application. The publisher can also access the Web Service access fees (from the metering and billing engine) that are to be charged to a particular consumer. The metering and billing engine accesses the licensing and logging databases to calculate the access fees to be charged to the consumer (see **Figure 1**).

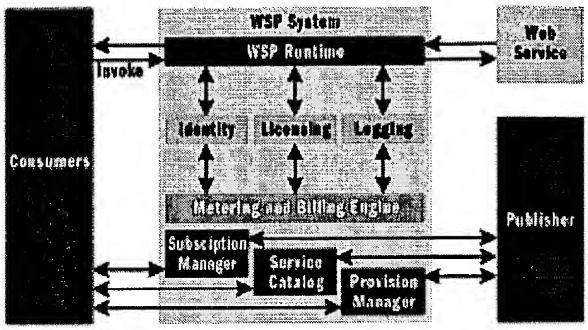


Figure 1 WSP Process Flow

Now let's take a look at the Web Service consumer process flow. A Web Service consumer uses the subscription manager to search and discover a Web Service. Upon discovering a Web Service that satisfies their business needs, the consumer completes the subscription process by providing information for the Web Service access security (such as their host name or IP address) and by choosing a licensing model that suits their needs.

Once the subscription process is completed, the Web Service is ready to be consumed. When a consumer invokes a Web service, the WSP runtime engine checks the identity and the licensing details and if they are satisfied, the service request is fulfilled and the usage details are logged in the logging database.

The Web Service usage logs now can be viewed by the Web Service consumer and the consumer can even view the access fees using the metering and billing engine. The consumer sees the same exact logs and access fees as the service publisher sees for this consumer.

Although the WSP system looks overly complicated at first glance, there are several advantages to using this system. For example, Web Services can be designed to perform only the business process tasks for which they are intended, so you really don't have to worry about implementing management, such as security, into the Web Service. Another advantage is that Web Service provision/federation is automated. Service monitoring and usage logs are available for every Web Service that is registered with the WSP system. Billing can be automated and sent to the subscribers periodically. Finally, service usage logs can be used to determine the quality of service (QoS) and hence the service level agreements (SLA).

Now let's look at how to build this system using the .NET Framework. I'm going to divide this task into two parts. First I'll build a generic provisioning framework, then I'll implement it for a real Web Service. I'm going to use C# for core classes (framework) and Visual Basic® .NET for implementation, and use a SQL Server 2000 database as the back end. In order to obtain the complete source code, please check the download for this article.

Building the WSP Framework

When I published a couple of Web Services on my online XML Web Services portal, I used to get a lot of SOAP traffic. Out of curiosity, I wanted to monitor the traffic, but I also wanted to limit some of the Web Services exclusively for my registered members. The WSP framework enables me to monitor and secure Web Services on my site using a process that is independent of my Web Services themselves.

The WSP framework has three architectural layers: the SOAP message interception layer (HTTP module), the SOAP message processing layer (WSP framework classes), and the data storage layer (SQL Server 2000).

SOAP Message Interception Layer

This layer consists of an HTTP module that intercepts every message received by the Web server. Since I'm only interested in SOAP messages, I need to distinguish them from all the other requests that are coming in.

In order to do this, I could just look for the SOAPAction header within the incoming message. Unfortunately, this method is not completely reliable because the SOAP specification makes a SOAPAction header optional.

There are two ways to handle this. I could look at the incoming HTTP request's target HTTP handler. As you probably know, HTTP handlers are responsible for processing the incoming HTTP Web requests. Each request type (or file extension) in .NET is mapped to a particular HTTP handler. The HTTP handler that handles requests to Web Services (ASMX files) is contained in the WebServiceHandlerFactory class, found in the System.Web.Services.Protocols namespace. So I will check for the HTTP handler's namespace to make sure it is being dispatched for a SOAP message (a Web Service request). Otherwise, I could check the file extension (.asmx) of the incoming HTTP request's URL.

Since I'm going to check for the HTTP handler, I need to do so after the handler has been dispatched for the current request. To do that, I need to wire up the onPreRequestHandlerExecute event, as shown here:

```
public class eSynapsHTTPModule : IHttpModule
{
   public void Init(HttpApplication httpApp)
    {

    httpApp.PreRequestHandlerExecute += new
        EventHandler(this.onPreRequestHandlerExecute);

    httpApp.PostRequestHandlerExecute += new
        EventHandler(this.onPostRequestHandlerExecute);
    }
```

And the following code shows the onPreRequestHandlerExecute function itself:

```
public void onPreRequestHandlerExecute(object o, EventArgs ea)
{
    HttpApplication httpApp = (HttpApplication) o;
    Type t = (httpApp.Context.Handler).GetType();
    if((httpApp.Context.Request.ServerVariables["HTTP_SOAPACTION"]
       != null) |
       (t.Namespace.ToString() == "System.Web.Services.Protocols"))

       {
         //This is a Web Service, do processing
```

SOAP Message Processing Layer

This layer consists of a set of nine classes that examine the incoming messages and perform various activities such as authorizing access to the service as well as denying unauthorized access (for expired subscriptions, for example) and logging the service usage details. Although **Figure 2** presents a simplified version of the actual framework process flow, it should give you a general idea of what functionalities the framework classes need to have.

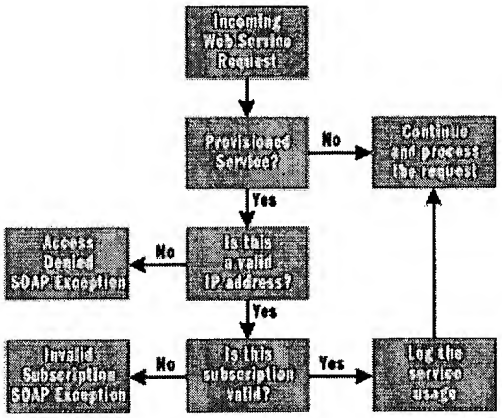


Figure 2 Framework Process Flow

Here's some detailed information on the functionality and implementation for each of the classes:

User class This class represents both service providers and service consumers in the WSP system. Each User class instance will have a unique ID and a flag that indicates whether the user is a provider, consumer, or both. If a user is a service provider, each instance of that user will be associated with corresponding entries in the service catalog with service provision details. If a user is a service subscriber, each instance of that user will be associated with related Web Service subscription details.

Service class An instance of a Service class represents a published or provisioned Web Service in the WSP system. Each Service class instance is associated with a provider, instance of a user class, and any provision details. This class is used in both subscription manager and provision manager applications.

ProvisionModel and ValidProvision classes These two classes together define a provision detail for a published Web Service. A ProvisionModel class defines valid provisions that are allowed in the system. A valid provision model could take one of three forms: free subscription, pay-per-use, or lease based. The ValidProvision class assigns each published service a provision model and pricing model. The provision manager application and WSP runtime mostly use these two classes.

Subscription class This class represents an instance of a subscription contract of a service consumer. An instance of this class links a service consumer to a Web Service mapped to a valid provision model. This class exposes methods that are useful for saving and updating Service subscription-related data. This class is used by both provision manager and subscription manager classes.

ValidIP class This class, which is used in the subscription manager, exposes methods that are required to implement IP-level security for Web Services in the framework. A subscription class should have an associated ValidIP instance, meaning each subscription should be mapped to a valid IP address with which the service is going to be accessed. If an incoming request's IP address does not match the IP address that was specified at the time of subscription, the service access will be denied.

Indeed, you can provide security to your Web Service in several ways. One method is to add a SOAP header containing the consumer's identity and password. This method works, but you need to share the identity and password if there are multiple users accessing a subscribed Web service. Otherwise, you need to create a subscription instance for each user who is going to use the Web Service. In addition to that, you need to add processing in the Web Service client to add a SOAP header in every outgoing message. If you would like to avoid all these issues, IP- or

domain-level security is a viable solution in a subscription-based Web Service.

Monitor class This is the only class that is accessed by the runtime application directly. The HTTP module creates an instance of this class for every incoming Web Service request. The Monitor class exposes functions that verify whether an incoming request is associated with a valid IP to access the Web Service. If an incoming request is a valid request with a valid subscription, the access to the service is granted and the usage is logged. If an incoming request fails to match the subscription or IP records, the Monitor class throws a SOAP exception with the appropriate error message.

This class uses the Service and ValidIP class. **Figure 3** shows how the Monitor class works.

BillingModel and BillingData classes These two classes implement the billing process in the framework. The billing functionality is usually comprised of the process of analyzing the service usage data and calculating the bills based on the provision and subscription models. Since this functionality is totally business-dependent and varies from company to company, they have not been implemented. However, if you want to take a look, you can still find them in the framework with basic template code.

The classes I discussed in this section provide the core functionality of the WSP framework. The source code for these classes is available in the download. Later in this article, I will show you how to leverage these framework classes by implementing a WSP system.

Data Storage Layer

As mentioned, I've relied heavily upon SQL Server 2000 and T-SQL in the WSP framework. A fine tuning of the stored procedures and database tables (such as indexing) is required because the incoming Web Service requests are being intercepted and processed. Any compromise at the database end will reflect on the Web Service performance and hence on the quality of the service.

The partial database design for the WSP framework is shown in **Figure 4**. As you can see, the WSDiscover table is the centerpiece of the data model. This table stores the Web Service details and can be linked to the UDDI by using the column UDDIKey.

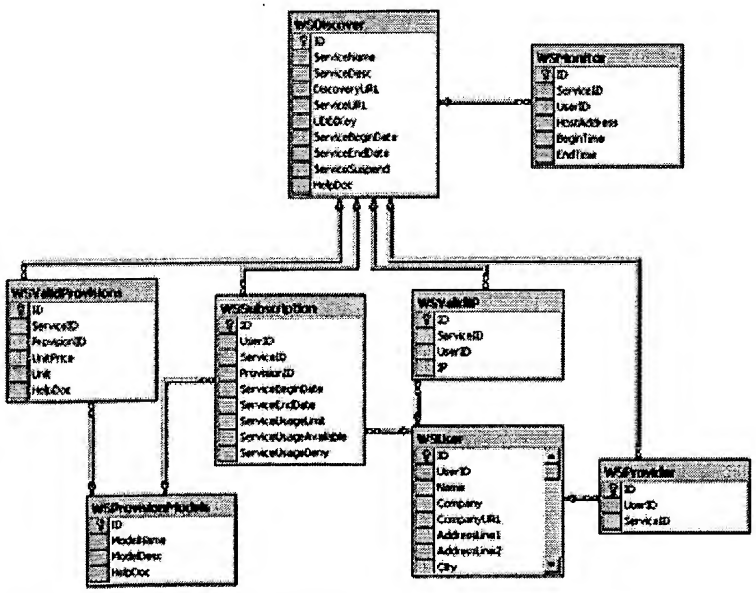


Figure 4 WSP Database Design

In addition to this database schema, a set of stored procedures is in place to work with the framework classes. Most of them implement simple save, update, and retrieve functionalities (a complete list of stored procedures is available in the source code download). However, I would like to discuss the most important stored procedure, usp_WS_IsValidSubscription_Monitor, which is used by the Monitor class to determine the validity of the subscription.

Figure 5 shows part of the usp_WS_IsValidSubscription_Monitor stored procedure. As you can see, the logic discussed in the diagram in **Figure 2** is partly implemented in this stored procedure. Mainly, the subscription validation check and service usage limit check are implemented here in order to gain overall system performance in the runtime.

Implementing the WSP System

Before I plunge into the implementation details of the system using the framework I outlined earlier, let's discuss a real scenario in which I need a WSP system and what I'm going to achieve with this implementation.

Suppose I have a Web Service which exists on my online XML Web Service portal. It allows users to search through my reference database for any Web Service-related topics. Anyone can type in the following URL (or create a Web Service client class using wsdl.exe) and access the Web Service at http://www.esynaps.com/WebServices/eSynapsSearch.asmx.

If I wanted to provide this Web Service exclusively for my registered members, I would need a solution that met several requirements. I should be able to prevent the general public from accessing the Web Service. My registered users, however, should be able to discover and subscribe to this service online. They should also be able to specify their own security credentials (IP address). That said, I should have the ability to deny the subscription, and my users and I should be able to verify the usage logs. Users should have the ability to subscribe to and cancel the Web Service whenever they want. Also, the solution should be generic enough for me to add and remove new services with ease. These requirements can apply to any Web Service whatsoever.

From the specifications I outlined, I need to implement the following applications: service provision manager, Web Service request interception, and service subscription management. I will review the implementation of each application. You can also take a look at an actual implementation of this system at http://www.esynaps.com/wsps.

Implementing Service Provision Manager

I have created a self-registration form for my users on eSynaps.com. When my users log onto the Web site, they will be presented with an option to register in the WSP system shown in **Figure 6**.

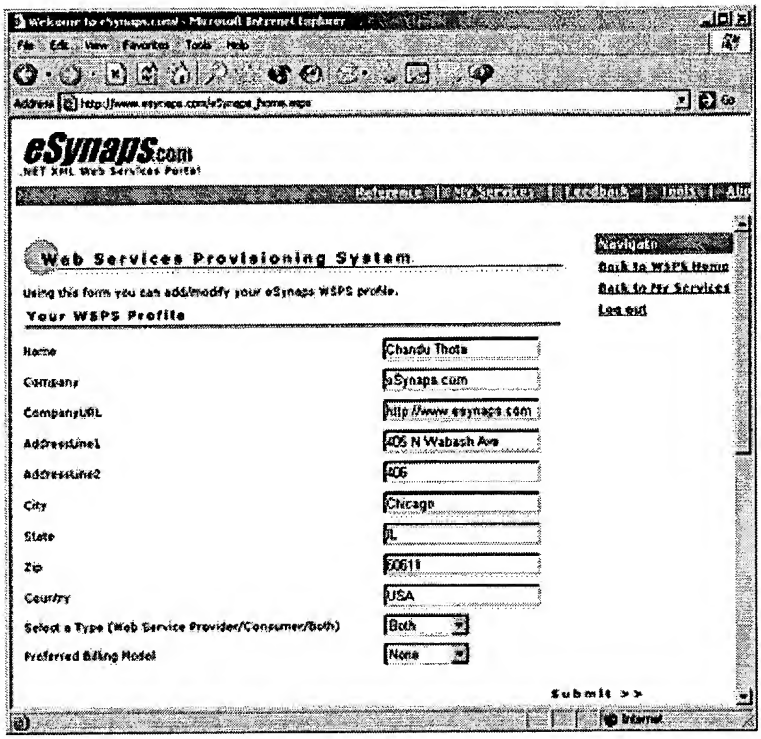


Figure 6 Self-registration Form

Successful completion and submission of the self-registration form will create a new user in the WSP system. The code behind this page is shown in **Figure 7**.

Upon completing the registration process the user should be presented with options based on their role in the WSP system. For example, a user who chooses to be a consumer should be given access to the subscription manager application. Similarly, a provider should be given access to the provision manager. If a user chooses to be both a provider and a consumer (as selected in **Figure 6**), they should be presented with both subscription and provision manager applications for their use.

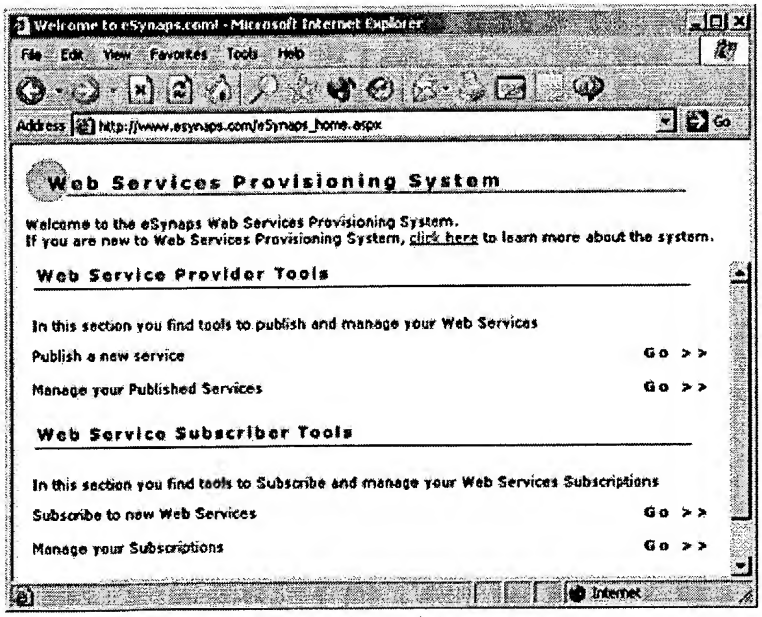


Figure 8 User as Service Provider and Consumer

Figure 8 shows both the subscription manager and provision manager applications displayed for the user who is registered as both a service provider and a service consumer. As you can see, the provision manager has two options: "Publish a new service" and "Manage your Published Services."

Publish a New Web Service

Publishing a new service involves two activities. First, you need to specify the service description, network endpoint (ASMX file location), and Web Services Description Language (WSDL) contract location. Second, you have to assign a proper provision model for the service. **Figure 9** shows the form that collects this information. As you can see from **Figure 9**, the service details and the provision details must be provided to publish the eSynapsSearch Web Service in the WSP system. The code that processes this form is shown in **Figure 10**. **Figure 11** details the SaveService method.

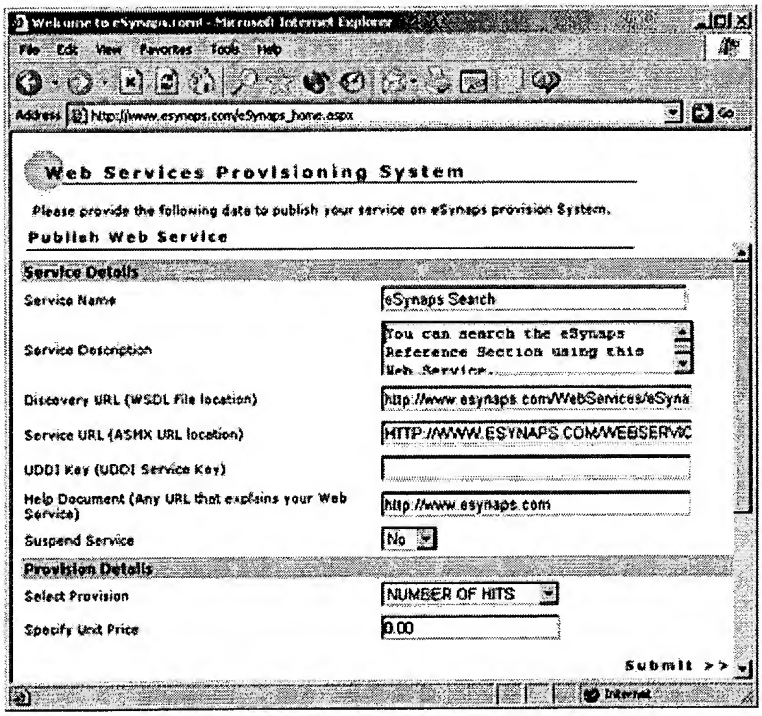


Figure 9 Specifying Service and Provision Details

Managing Existing Services

The second stage of implementing the provision management system is to provide the ability to manage the published Web Services. For that purpose, you can leverage the Service class from the WSP framework. Using a Service class, you can browse all of the published services and provide functionality to update the service. If you click the Manage link on the page, it will take you to the service publish/update screen shown in **Figure 9**.

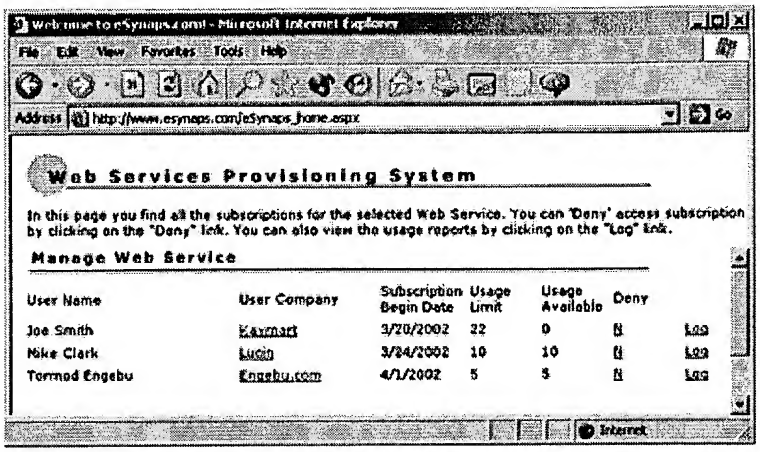


Figure 12 Manage Web Service

Here's the code behind the Manage Web Service page that is shown in **Figure 12**:

```
Private Sub Page_Load(ByVal sender As System.Object, ByVal e As _
    System.EventArgs) Handles MyBase.Load

    If Page.IsPostBack Then
    ...
    Else
    ...
      Call BrowseServicesByProvider()
    End If
End Sub
```

The BrowseServicesByProvider method is shown in **Figure 13**. As its name implies, it lets you browse services by provider info.

The Reports link allows you to see all consumers that are subscribed to the service. In my case, the screen in **Figure 12** appears when I click the Reports link.

As you can see in **Figure 12**, there are three users subscribed to the Search Web Service. The options for the service provider on this screen include denying service for a particular user and viewing the usage logs. But before I move on to the usage logs, let's see what is going on behind the scenes.

In this ASPX page, I'm using the Subscription class. This method explains how to retrieve users based on a service subscription:

```
Dim dr As SqlDataReader
'Declare a Subscription class
Dim oSub As New WSPF.Subscription()

'Then browse with Service ID for all the Subscriptions!
Call oSub.Browse(CType(Request.Params("ServiceID"), Integer), _
    Subscription.Type.ByService, dr)
'Then bind to the Data Repeater
rp.DataSource = dr
rp.DataBind()
```

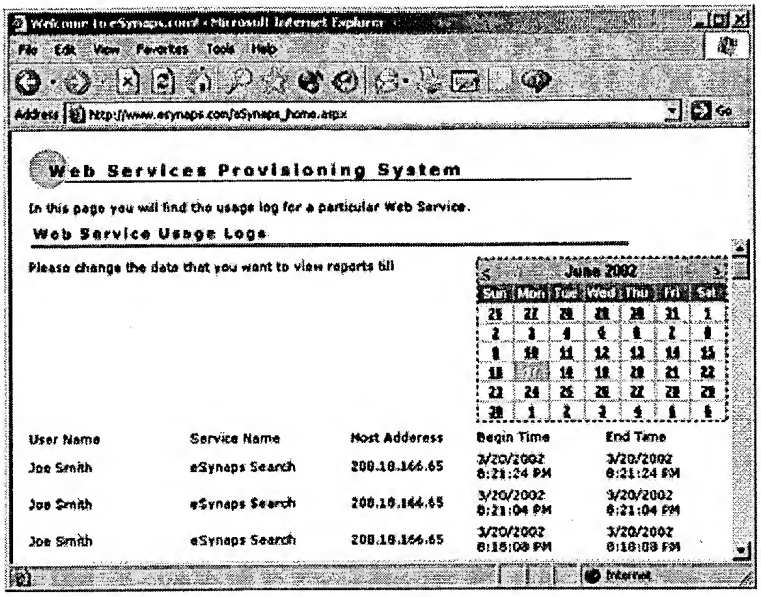


Figure 14 MWSMonitor Table

Now, let's look at the usage logs portion of the provision management application. When you click the Log link shown in **Figure 12**, you will see the screen shown in **Figure 14**. This page displays the contents of the MWSMonitor table. The codebehind for this page is shown here:

```
'Declare a WSPF Monitor object
Dim oMon As New WSPF.Monitor()
Dim dr As SqlDataReader

'Browse by Service ID and User ID
oMon.Browse((CType(Request.Params("ServiceID"), Integer)), _
    oSub.UserID, fromdate.SelectedDate.AddDays(1), dr)

'Bind to the Data Repeater
  rp.DataSource = dr
  rp.DataBind()
```

At this point I know that you're probably wondering, "how I got this data into Monitor Tables." The next section answers this question.

Web Service Request Interception

I have now successfully published a Web Service in the WSPS. But does that mean that the public's access to my Search Web Service is restricted? No. The process is not complete until you configure the Web Ser-vice request interception layer or the WSP runtime application.

As I discussed at the beginning of this article, the SOAP message interception layer (the HTTP module I designed in the first part of the article) does the actual security check and service usage metering and logging.

Figure 15 shows the implementation of the HTTP module. The onPreRequestHandlerExecute method is executed before the HTTP handler starts executing the Web Service.

The onPostRequestHandlerExecute method (see **Figure 16**) will be executed from the HTTP module when the HTTP handler finishes processing the Web Service. It calls the Save method on the Monitor class that was created in the method shown in **Figure 15**. When this function calls the save function on the Monitor class, the begin time and end time of the service request are saved into the WSMonitor table. That data is then used to generate the usage logs for the services.

To be able to put this HTTP module to work so that it can intercept the incoming Web Service requests, you need to add the following configuration setting to your web.config file:

```
<system.web>
  •••
  <httpModules>
    <add name="httpmod" type="eSynaps.WSPS.HTTP.eSynapsHTTPModule, _
        eSynaps.WSPS.HTTP" />
  </httpModules>
  •••
</system.web>
```

Now let's take a look at the state of the Web Service. When a user who did not register himself in the WSP system tries to access the Web Service, a SOAP exception will be thrown and a message displayed indicating that access to the Web Service is denied for unregistered users. Now the user will have to subscribe to this Web Service if they want to use it.

Service Subscription Manager

Upon receiving an error message (denying access to the service), a new user might log onto the site and subscribe as a consumer. A subscriber will have the option to subscribe to a Web Service or manage existing subscriptions. The subscription screen for the consumer uses the Service class from the WSP framework to display all available services. The user then views the details of the two available services and subscribes. The code for subscribing a new user to a Web Service is shown here:

```
'Declare a new Subscription class
Dim oSub As New WSPF.Subscription()

'Set the values
oSub.UserID = CType(Session("WSUID"), Integer)
oSub.ServiceID = CType(ServID.Value, Integer)
oSub.ProvisionID = CType(ProvID.Value, Integer)
oSub.ServiceBeginDate = System.DateTime.Now()

'Set the Usage limit-some random number for testing
oSub.ServiceUsageAvailble = 10

•••

'And call Save!
lSub = oSub.Save()
```

Once subscribed to the Search Web Service, the user has a usage limit of 10 hits. Next, the user will be presented with a screen to specify the IP address from which they are going to access the Web Service. Upon providing the right IP address and clicking the Submit button, the IP address will be saved in the WSP system tables using the ValidIP class from the WSP framework:

```
Dim oValidIP As New WSPF.ValidIP()
•••

'Set the Values

oValidIP.ServiceID = CType(ServID.Value, Integer)
oValidIP.UserID = CType(Session("WSUID"), Integer)
oValidIP.IP = IP.Text

'Call Save
lVIP = oValidIP.Save()
```

Now the user can access the Search Web Service 10 times.

What happens if this user tries to exceed the usage limit? On the eleventh request to this Web Service, access to the service will be denied.

The Monitor class from the WSP framework will take care of the limit check. If the user still wants to use this Web Service, he has to renew the Web Service subscription.

Don't forget that the user also has access to the service usage logs in the Manage Subscriptions area. The service usage logs available for service consumers look just like the ones I showed in the provider's case.

Conclusion

The Web Services provisioning framework I built here is a powerful utility application that works well on a Web server. However, you can also implement the same system in an application service provider model by making the Monitor class remotable. You could try to expose the Monitor class as a Web Service, but remoting proves to be more efficient in this case. You could also implement your own security model and billing model in this system, to fit your particular needs.

Whether in a business-to-consumer case or in a business-to-business scenario, a properly designed Web Services provisioning system will allow Web Services to concentrate on the business processes and implementations rather than the infrastructural aspects of the Web Service technology.

For related articles see:
XML Web Services Developer Center
SQLXML Server Developer Center
For background information see:
The ASP Column: HTTP Modules
House of Web Services: The Continuing Challenges of XML Web Services

---

Chandu Thota works as a Technical Lead for a Chicago-based channel management software company. He has published several articles related to .NET and founded an online .NET XML Web Services portal, http://www.esynaps.com. He is also a coauthor of the books *Understanding the .NET Framework* and *Building an ASP.NET Intranet*, both published by WROX Press.

From the December 2002 issue of MSDN Magazine.
Get it at your local newsstand, or better yet, *subscribe*.

Back to top **QJ:** 021207

Print E-Mail Add to Favorites

How would you rate the quality of this content?

1 2 3 4 5 6 7 8 9
Poor ○ ○ ○ ○ ○ ○ ○ ○ ○ Outstanding

Tell us why you rated the content this way. (optional)

Submit

Average rating:
8 out of 9

1 2 3 4 5 6 7 8

15 people have rat

Manage Your Profile | Legal | Contact us | MSDN Flash Newsletter

©2004 Microsoft Corporation. All rights reserved. Terms of Use | Trademarks | Privacy Statement

Search for: within All of dW Search
Search help
IBM home | Products & services | Support & downloads | My account
developerWorks > Web services
developerWorks

Web services provisioning

PDF e-mail it!

Understanding and using Web Services Hosting Technology

Mike Polan (polan@ca.ibm.com)
WebSphere Hosting Architect, IBM Electronic Commerce division
1 January 2002

Provisioning Web services is a keystone to operating pay-per-use Web services between businesses. It is a complex mixture of service authentication, enrollment, metering, billing, and managing operations that control the behavior of a Web service during use, whether within your own company or between business partners. This paper examines the Web Services Hosting Technology (WSHT) package released through alphaWorks that handles this major task.

Related content:
Fee-based Web services, Part 1
Fee-based Web services, Part 2
Fee-based Web services, Part 3

Subscriptions:
dW newsletters
dW Subscription (CDs and downloads)

Introduction

Web services architecture simplifies the creation of distributed applications by leveraging the pervasive nature of the Internet. Organizations can now have cheap, open pipelines to their customers and partners. This allows the creation and integration of automated processes between business entities, as well as the creation of hosted services that can be used as parts of distributed business processes.

For example, a business may implement a purchasing system. In addition to tracking orders and managing the workflow needed for internal approvals, this system could automatically discover, solicit bids, and place orders through external suppliers. It could discover and arrange financing and shipping for orders, and track order status. By leveraging Web services, the purchasing system remains within the realm of control of the organization managing the process, yet can delegate to external organizations for the services needed to complete the process.

While much work remains to realize this goal, the infrastructure is beginning to emerge. The Internet provides the connectivity; the Web services architecture provides the mechanisms for exchanging data. Standards bodies will continue to refine service interface specifications to further reduce integration effort.

Service providers or businesses that publish Web services for internal or external use will need management functions to provision and control these services. Dan Gisolfi described the requirements identified by the IT department of the fictional "Trumpet" company in his Web services architect column (see Resources). An alphaWorks package, "Web Services Hosting Technology" (WSHT V1.0) demonstrates how the concerns of the Trumpet IT department can be partially addressed. This paper discusses using WSHT in that environment, and how the remaining requirements might be addressed in the future.

Hosting scenarios

The key areas of focus of WSHT on Web services provisioning are as follows:

1. Tooling to allow the publishing of Web services into a catalog referenced by the subscription system.
2. A subscription and provisioning system for Web services.
3. Metering and billing of Web services relative to the contract associated with subscriptions.
4. A management system for the subscriptions, provisioning, metering, and billing components.

We begin by looking at the two key scenarios:

1. The hosting of Web services built and running on a local application server within the corporate intranet.
2. The hosting of Web services built and running on an application server outside of the corporate intranet.

The distinction between the two scenarios is that the first is a simple hosting scenario, while the second could be described as a brokering or wholesaling scenario. For further information on this distinction, you should read "Applying Web services to the application service provider environment" (see Resources).

The goal in both cases is to allow the management of Web services in a fashion that separates the problem domain of the service itself from that of the subscription, provisioning, and management system; in many cases, the author of the service will not need to consider the needs of the provisioning system when composing the service.

The development and deployment aspect would involve of the following steps:

1. A developer uses a Web services toolkit and unit test environment to create and test the Web service.
2. The developer provides the provisioning hooks to allow a provisioning system to programmatically enroll a new user of the service (if needed).
3. The service provider creates a rating package for the rating and billing engine (using tools provided by those applications).
4. The Web service is deployed onto the service provider's production server.
5. The Web service definition (WSDL) is deployed to UDDI or some other WSDL registry.
6. The service provider creates an offer (an offer contains one or more WSDL registry listed services which are associated with the rating package).
7. The service provider deploys the offer to an online catalog referencing the services catalog and the rating package.

The enrollment and subscription aspect:

1. A potential subscriber (consumer) discovers the Web service(s) either via the registry referencing the catalog or vice versa.
2. The new subscriber enrolls (establishes an account) with the service provider if not already a member.
3. The subscriber selects to subscribe to the new Web service.
4. The service is automatically provisioned for the new subscriber.
5. The subscriber downloads the assets necessary to work with the Web service, composes, and deploys a client application.

And the usage aspect:

1. The client application uses the Web service, usage is metered and reported by the Web services provisioning infrastructure.
2. Usage events are passed to the rating engine, which generates invoice lines for the billing system.
3. Both the service provider and the subscriber access the billing system for the current usage charges.
4. The service provider uses the management UI to control access to the Web service.

Each of these aspects involves the provisioning system at some level. The development and deployment aspect requires building an interface to the provisioning system; the enrollment and subscription builds the provisioning system; and the usage aspect makes use of the provisioning system. Thus, the enrollment and provisioning system, and the systems that support it, is the key to successfully hosting enterprise-level Web services.

Web Services Hosting requirements

To support Web services hosting on an enterprise scale, you will need a number of important components. This section builds on the requirements identified for hosting Web services that were introduced in "Metering and accounting for Web services"(see Resources).

Catalog of services

To manage Web services for hosting, the services may need to be associated with an offer or deal. An offer links Web services to an appropriate rating package used for determining usage fees (billing). The offer and associated services are published to a catalog and are then available for discovery and subscription by the consumer.

A rating package is a set of rules that associates subscription and usage meter events with a fee. The rating engine evaluates these events against the identified rating package, feeding invoice lines into a billing system. A billing system associates the invoice lines with the proper account, as well as a number of other functions such as tracking invoices and accepting payments.

Once defined, a publishing step makes the assets necessary for the consumption of a Web service available to the consumer. The offer information associated with the services is published to the catalog service and is referenced by the subscription service.

Enrollment and subscription

Enrollment is the process of making the hosting system aware of the subscriber identity. This identity is needed to later associate the use of a Web service with the consumer of that service. Often the enrollment and subscription steps are combined.

Subscription is the act of adding a user to the list of consumers of a given Web service. The consumer selects the desired service by choosing an offer containing the desired service(s) and identifying the terms of use in a contract. Subscription may require payment of a setup fee. Once subscribed, any assets necessary for the operation (consumption) of the service are made available to the consumer.

Provisioning

Provisioning is the act of preparing the system for the use of a service by a consumer. It involves preparing both technical and business aspects for supporting the consumers activities.

Web service provisioning

Services may be anonymous or provisioned. Anonymous services operate without requiring the identity of the consumer at run time. An example of an anonymous service would be the retrieval of a stock price. The authentication, authorization, and metering of a Web service is delegated to the hosting system, so anonymous services typically need no additional provisioning.

A provisioned service is one that requires association of a user account and/or other information with the service. For example, the retrieval of an account balance from a user portfolio would be a provisioned service, as the account balance to retrieve must be determined. Additional access control is required for fine-grained protection of the assets associated with that service.

Each provisioned service needs some extensions to automate the enrollment and subscription process. Typically these are APIs built into the service, and need only to be mapped to the provisioning system functions through a service agent. The provisioning system will then drive these functions through the agent at subscription time.

Supporting systems

A number of systems, present in any distributed system design, are useful as well when hosing Web services, and must be provisioned, as shown in Table 1.

Table 1. Provisioning support systems

System	Description
Member	A common repository to allow the sharing of user profile information between services. The member repository is populated during the enrollment step. Ideally profile management is self-service, the management functions being provisioned like any other service as part of an offer.
Authentication	This establishes the identity of the invoker of a service at run time. The authentication system is typically provisioned during enrollment.
Authorization	This system stores and reports on policies and permissions to determine authorized access to a service and prevent unauthorized access to components or services. The authorization system would be provisioned at subscription time, and consulted at service run time.
License management	This system assists the service provider in ensuring access to services are properly controlled under the terms of the license agreements for that service. License agreements may be to the service provider and the end user, or between the service provider and the supplier of the service. The license management system would be provisioned at the time the service was installed, and

	consulted at service run time.
Rating system	The rating system is used to determine the fees to be applied to meter events as they occur. The rating system consists of a rating engine, and a rating package for each consumer agreement (rating packages can be shared). The rating package would be created (or referenced) at service subscription time. The rating engine would typically process meter events (to determine fees) post service execution, although some implementation may require processing at service run time.

Contracts

A contract can be used to hold the information that links the consumer of a Web service to that service, to the terms of the service (including the rating package), and to the billing account. Authentication and identity systems provide the context used to discover the appropriate contract when the service is invoked, along with the invoked service itself. The contract context is used for metering information, by the Service Level Agreement (SLA) and by the Quality of Service (QoS) management systems. The hosting system creates and associate consumers, the services, SLA, QoS, and the billing account upon subscription.

In a hosted environment, attempts to invoke hosted Web services will be denied if a valid contract to use those services does not exist.

Metering, rating, and billing

The accounting process for provisioning has several subfactors of metering, logging, rating, and billing.

Metering

Use of the Web service by the consumer must be metered if you need usage-based billing. The system also collects enrollment, subscription and provisioning events, and can use this information to determine subscription fees.

Usage metering without imposing additional design requirements on the service itself is accomplished by the instrumentation of the Web services run time. Web services authors may further instrument their services to provide additional metering events.

Logging

Meter events are written to a common log for later processing by the rating engine. Metering and meter event logging may occur both at the server and the client sides of the Web service, allowing later reconciliation.

Rating

The rating engine translates meter events into invoice items passed to the billing system using the assigned rating package. The rating engine determines charges based on usage and subscription (time-based). Often the rating engine is part of a larger billing system.

Billing

The billing system accepts invoice information from the rating engine and associates it with the correct user account. The billing system provides information to allow the retrieval and payment of billing information by the consumer, reconciliation in the case of disputes, and the correct disbursement of payments to the provider's suppliers (who in turn may be service providers offering wholesale services to the original provider).

Run time for Web services Hosting

At run time, the various supporting systems again come into play as shown in Table 2.

Table 2: Hosting support systems at run time

System	Run-time procedures
Member	Services may need to retrieve member information from this common repository.
Authentication	Establish and validate the identity of the invoker of a service.
Authorization	Ensure users access only those services to which they are entitled.
Metering	The Web services infrastructure will record usage events to the logging system.

Logging	A common logging system will capture audit and meter records for later processing.
License management	Ensure the components of the system and services are used in a manner consistent with the terms of any license agreements that apply.

When a service is invoked, the Web services infrastructure will attempt to discover a contract to apply to this invocation. The contract would contain references to the information under which the service would be processed. At a minimum, the existence of a contract would serve as an authorization check; without a valid contract, the service cannot be invoked. The contract may also identify the rating package that should be used when processing meter events generated by this invocation, and may identify QoS and SLA parameters to be used when dispatching the service.

Management

The employment of self-serve profile management functions would lower the cost of operations of the service provider, and would empower consumers to manage their organization and personal data.

Management functions of the authentication and authorization systems are typically provided by those systems themselves.

Catalog publishing tools would be used to manage service and offering information in the catalog, while the hosting system would provide contract and subscription management.

Typically the billing system provides custom management function. The billing system may also provide subscriber interface components that would be integrated into an end user portal.

Web Services Hosting Technology

Web Services Hosting Technology (WSHT) is a new package released on alphaWorks that implements all the major functions of provisioning. The following is an overview of WSHT and how it works.

Components and function

The major components of WSHT are the portal, the offer catalog, the enrollment and subscription system, the run-time extensions, and the rating, billing, and accounting system.

WSHT portal

The WSHT package addresses many of the requirements for hosting Web services. It is not intended for production, so the enrollment, subscription, and administrative functions have been combined into a single portal for convenience. The portal provides access to:

- Catalog build and management functions.
- User and subscription management.
- Billing account status.
- Enrollment functions.
- Subscription functions.
- Entitlement lists.
- A Web services test interface.

Offering creation and offer catalog

WSHT provides a tool for creating and managing subscription elements known as offers. An offer associated one or more Web service from a UDDI registry with a rating package used for billing.

Offers are presented to enrolled users in a simple catalog as part of the enrollment and subscription operation.

Enrollment and subscription

A simple enrollment system is provided based on the WSTK Identity service. To enroll, a user simply selects a unique user id and provides a password.

A subscription interface allows the enrolled user to select from a list of available offers. Acceptance of the terms of the offer results in the creation of a contract used upon service invocation.

Run-time extensions
The basic Web services run time is extended in the WSHT environment. The run time has been instrumented to record audit and meter records. The basic WSTK identity service is used for authorization (if a user exists and the password matches they are authorized to use the system). The WSTK contract service is used to establish whether the user of a service has a valid contract in place, the contract identifier is then referenced on the meter records.

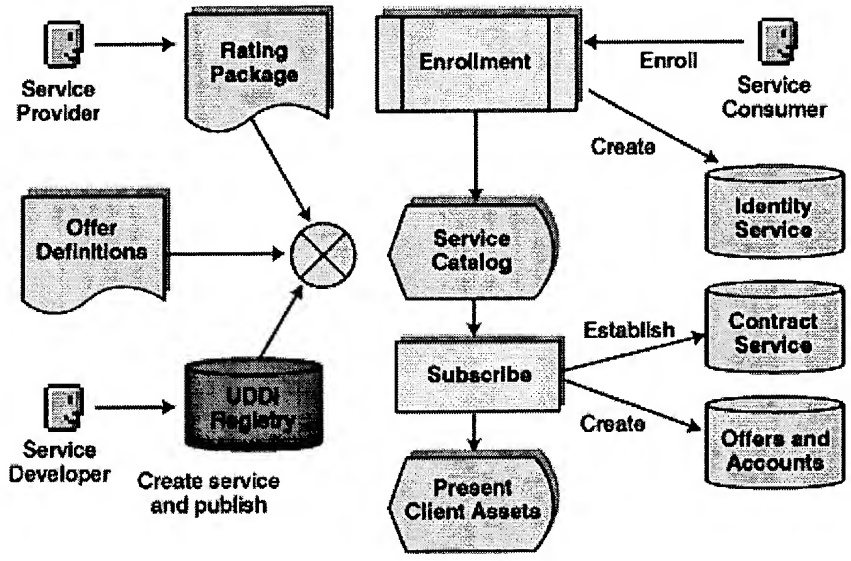
Rating, billing, and accounting
A lightweight rating engine periodically processes meter records found in the audit and meter log, passing invoice records to a lightweight billing system. The billing system associates invoice records to subscriber accounts. A user interface is provided to allow the viewing of consolidated subscriber bills.

Operating WSHT

Operating WSHT is a two step process; first, to establish the service with the provisioning system; and second to bind the service consumer client to the provisioning system.

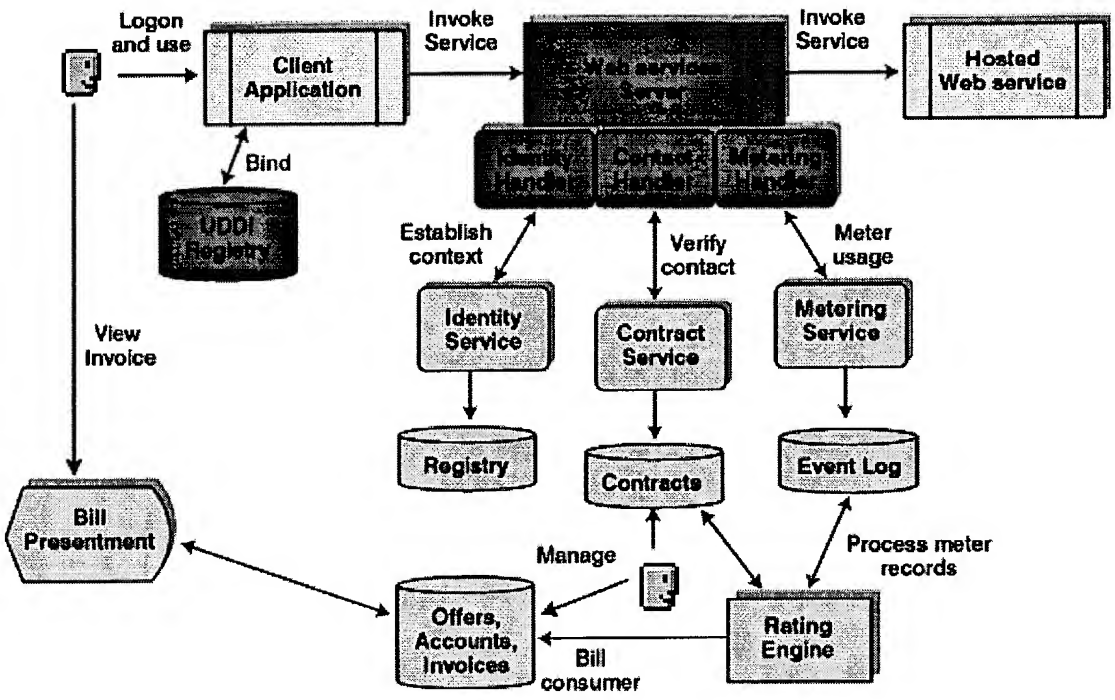
WSHT publish, enrollment, and subscription
Figure 1 shows how the service developer publishes the service definition to a UDDI registry. The service provider matches the service to a rating package as an offer, and creates a catalog entry. The consumer enrolls with the service provider, selects the service desired from the catalog, and subscribes to the service. The client assets are downloaded by the consumer and used in the development of the client application invoking the service.

Figure 1. Establishing a service with the provisioning system.



WSHT run time
At run time (see Figure 2), the client application binds to the service and invokes it using an appropriate mechanism for reporting the identity established in the enrollment phase (see Figure 1). The Web services server invokes the identity handler to verify the consumer identity, and then the contract handler to verify a contract exists. The metering handler is then called to record the service invocation start, and later, its completion. The rating engine processes events found in the event long, determines the charges and creates invoice items in the billing system. The consumer uses the service portal to view usage charges.

Figure 2. Run-time client application binding and services interaction



WSHT configurations

There are two configurations for WSHT to support different implementations of Web services protocols: simple and gateway.

Simple

The simple configuration assumes the use of the handler system in the Apache SOAP 3.0 (Axis) protocol implementation, for establishing the identity of the user of a service. The service client inserts the user id and password into the request. The Web Services ToolKit 3.0 server (see Resources) retrieves the identity and uses it to discover the applicable contract under which the service will be invoked.

Gateway

The Web Services Gateway (WSGW) can be used for more complex scenarios, such as:

- Support for non-Axis clients.
- Support for non-Axis hosted services. In particular, the gateway can be used in the "wholesale"scenario, where a service provider is a broker or portal for external services.
- Support for WSIF-invoked internal services, including RMI-IIOP

The WSGW is instrumented for hosting by the addition of interceptors which function in the same fashion as the Axis handlers described above. The gateway will allow the establishment of consumer identity without reliance on the Axis client handlers. For example, WSHT will support HTTP authentication when using the WSGW.

Conclusion

WSHT demonstrates how the Web Services ToolKit and the Web Services Gateway can be used to provide the core functions needed to operate a pay-for-use Web service hosting system. Contracts can be used to link services and consumer identities to rating packages. The rating packages convert usage meter records into invoice records and associated with a consumer account.

Most importantly, any service, local or remote, can be provisioned and managed by the WSHT hosting support without requiring changes to the Web service itself. Architectures such as this one will simplify the acceptance of Web services into service provider environments.

Resources

- Dan Gisolfi's column issues on Fee-based Web services: Part 1, Part 2, Part 3.
- Dietmar Kuebler and Wolfgang Eibach on Metering Web services
- Download the Web Services ToolKit from IBM alphaWorks.
- Download the Web Services Hosting Technology package from IBM alphaWorks.

About the author

Mike Polan is a WebSphere Architect currently focused on application and service provisioning, in particular Web services provisioning development. He was a development manager for WebSphere Commerce Suite and lead development teams on VisualAge C++ and VisualAge Java. He can be reached at polan@ca.ibm.com.

PDF e-mail it!

What do you think of this document?

○ Killer! (5) ○ Good stuff (4) ○ So-so; not bad (3) ○ Needs work (2) ○ Lame! (1)

Comments?

Submit feedback

developerWorks > Web services

developerWorks

About IBM | Privacy | Terms of use | Contact

PORTAL
US Patent & Trademark Office

Subscribe (Full Service) Register (Limited Service, Free) Login

Search: The ACM Digital Library The Guide

+provision* +<paragraph> +web +<paragraph> +service* SEARCH

THE ACM DIGITAL LIBRARY

Feedback Report a problem Satisfaction survey

Published before October 2001
Terms used **provision paragraph web paragraph service**

Found **95** of **114,536**

Sort results by relevance

Display results expanded form

Save results to a Binder

Search Tips

Open results in a new window

Try an Advanced Search

Try this search in The ACM Guide

Results 21 - 40 of 95 Result page: previous 1 **2** 3 4 5 next

Relevance scale

21 The travails of visually impaired web travellers

Carole Goble, Simon Harper, Robert Stevens

May 2000 **Proceedings of the eleventh ACM on Hypertext and hypermedia**

Full text available: pdf(229.51 KB) Additional Information: full citation, references, citings, index terms

Keywords: hypertext, mobility, navigation, travel, usability, visual impairment, web

22 Annotea: an open RDF infrastructure for shared Web annotations

José Kahan, Marja-Ritta Koivunen

April 2001 **Proceedings of the tenth international conference on World Wide Web**

Full text available: pdf(271.46 KB) Additional Information: full citation, references, citings, index terms

Keywords: RDF, World-Wide Web, XML, XPointer, annotations, metadata, semantic web

23 Web-based simulation: revolution or evolution?

Ernest H. Page, Arnold Buss, Paul A. Fishwick, Kevin J. Healy, Richard E. Nance, Ray J. Paul

January 2000 **ACM Transactions on Modeling and Computer Simulation (TOMACS)**, Volume 10 Issue 1

Full text available: pdf(90.00 KB) Additional Information: full citation, abstract, references, citings, index terms, review

The nature of the emerging field of web-based simulation is examined in terms of its relationship to the fundamental aspects of simulation research and practice. The presentation, assuming a form of debate, is based on a panel session held at the first International Conference on Web-Based Modeling and Simulation, which was sponsored by the Society for Computer Simulation during 11-14 January 1998 in San Diego, California. While no clear "winner" is evident in this debate, the ...

Keywords: Java, digital objects, distributed modeling

24 The LATEX legacy: 2.09 and all that

Chris Rowley

August 2001 **Proceedings of the twentieth annual ACM symposium on Principles of distributed computing**

Full text available: pdf(956.36 KB) Additional Information: full citation, abstract, references, index terms

The second edition of The Manual [23] begins: `LATEX is a system for typesetting documents. Its first widely available version, mysteriously numbered 2.09, appeared in 1985.'

It is too early for a complete critical assessment of the impact of LATEX 2.09 because its world-wide effects on many aspects of many cultures, not least scientific publication, remain strong after 15 years—and that itself is significant in a technological world where a mere 15 months of fame can make *an* ...

25 Measuring information quality of web sites: development of an instrument

Pairin Katerattanakul, Keng Siau

January 1999 **Proceeding of the 20th international conference on Information Systems**

Full text available: pdf(134.94 KB) Additional Information: full citation, references, citings, index terms

26 Extending support for contracts in ebXML

James Cole, Zoran Milosevic

January 2001 **Australian Computer Science Communications**, Volume 23 Issue 6

Full text available: pdf(947.74 KB) Publisher Site Additional Information: full citation, abstract, references, citings

This paper describes our proposal for extending the current ebXML standard - to provide more comprehensive support for electronic contracts. The paper first presents the current status of major B2B initiatives, focusing on their support for electronic contracts. Having found the ebXML meta-model to provide a suitable contractual foundation, we examine the full extent of the requirements for supporting electronic contracts, and propose extensions to the ebXML meta-model to facilitate these requir ...

27 Congestion-dependent pricing of network services

Ioannis Ch. Paschalidis, John N. Tsitsiklis

April 2000 **IEEE/ACM Transactions on Networking (TON)**, Volume 8 Issue 2

Full text available: pdf(339.88 KB) Additional Information: full citation, references, citings, index terms

Keywords: Internet economics, dynamic programming, loss networks, revenue management

28 Creating an HTML help system for web-based products

Laura Rintjema, Kara Warburton

September 1998 **Proceedings of the 16th annual international conference on Computer documentation**

Full text available: pdf(770.53 KB) Additional Information: full citation, references, citings, index terms

Keywords: hypertext information system, information architecture, navigation, task-oriented help

29 A case of academic plagiarism
Ned Kock
July 1999 **Communications of the ACM**, Volume 42 Issue 7
Full text available: pdf(174.73 KB) html(43.07 KB) Additional Information: full citation, references, citings, index terms

30 Harnessing technology for effective inter- and intra-institutional collaboration: report of the ITiCSE '97 working group on supporting inter- and intra-institutional collaboration
Douglas Siviter, Marian Petre, Bruce Klein
October 1997 **ACM SIGCUE Outlook**, Volume 25 Issue 4
Full text available: pdf(2.66 MB) Additional Information: full citation, abstract, references, index terms

The computer science discipline is well poised to provide leading examples of harnessing communications and computer technologies in order to encourage collaborative practices both within and between institutions. Students, academics, and institutions all potentially have access to their counterparts world-wide. This provides endless opportunities for sharing knowledge, accessing scarce expertise, making effective re-use of limited resources, collaborating to attract funding and influence polici ...

31 Harnessing technology for effective inter- and intra-institutional collaboration (report of the ITiCSE '97 working group on supporting inter- and intra institutional collaboration)
Douglas Siviter, Marian Petre, Bruce Klein
June 1997 **The supplemental proceedings of the conference on Integrating technology into computer science education: working group reports and supplemental proceedings**
Full text available: pdf(145.01 KB) Additional Information: full citation, references, citings, index terms

32 Fair queuing for aggregated multiple links
Josep M. Blanquer, Banu Özden
August 2001 **ACM SIGCOMM Computer Communication Review , Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications**, Volume 31 Issue 4
Full text available: pdf(184.49 KB) Additional Information: full citation, references, citings, index terms

33 Regulation of technologies to protect copyrighted works
Pamela Samuelson
July 1996 **Communications of the ACM**, Volume 39 Issue 7
Full text available: pdf(671.52 KB) Additional Information: full citation, references, citings, index terms, review

34 Awareness and the WWW: an overview
Olivier Liechti
December 2000 **ACM SIGGROUP Bulletin**, Volume 21 Issue 3
Full text available: pdf(1.47 MB) Additional Information: full citation, abstract, references

The notion of awareness has received a lot of attention in the CSCW literature for quite some time now. Because it cannot be very precisely and uniquely defined, this notion covers a range of issues and is critical in very different situations. This is also true in the particular context of the WWW, where awareness has more than one facet. One objective for this paper is to give an overview of the field, by reviewing different awareness categories and by showing how they relate to Web-based syst ...

Keywords: CSCW, WWW, activity space, awareness, contextual awareness, group awareness, implementation platform, peripheral awareness, workspace awareness

35 Paradigms 2: ETEL: a newspaper-based distributed information system

Michel Banâtre, Valérie Issarny, Frédéric Leleu

September 1996 **Proceedings of the 7th workshop on ACM SIGOPS European workshop: Systems support for worldwide applications**

Full text available: pdf(669.55 KB) Additional Information: full citation, abstract, references

The ever-growing number of on-line multimedia services relating to application fields as diverse as commercial, entertainment, and educational fields calls for distributed information systems that help the end-user accessing available services. Most popular information systems, also qualified as resource discovery services, are those provided in the Internet such as the World-Wide Web (Www), which is becoming the dominant Internet Resource [4]. Other systems include those based on the use of pri ...

36 Architectural framework modeling in telecommunication domain

Giulio Fregonese, Alessandro Zorer, Giovanni Cortese

May 1999 **Proceedings of the 21st international conference on Software engineering**

Full text available: pdf(1.12 MB) Additional Information: full citation, references, citings, index terms

Keywords: architectural patterns, design patterns, distributed systems, domain analysis, network and service management, network traffic data analysis, object-oriented framework, software architecture, software reuse

37 Other impairments and rehabilitation technologies: Universal access to mobile telephony as a way to enhance the autonomy of elderly people

Julio Abascal, Antón Civit

May 2001 **Proceedings of the 2001 EC/NSF workshop on Universal accessibility of ubiquitous computing: providing for the elderly**

Full text available: pdf(759.75 KB) Additional Information: full citation, abstract, references, index terms

The rise of mobile telephony has opened a vast diversity of new opportunities for older people with different levels of physical restrictions due to ageing. Mobile technology allows not only ubiquitous communications but also anytime access to some services that are vital for elderly people's security and autonomy. Nevertheless, with the numerous advantages, remote services can also introduce important social and ethical risks for this group of users. This paper tries to analyse the novelties th ...

Keywords: assistive technology, mobile communications, ubiquitous computing, universal design

38 The "growing up" of HyperBraille—an office workspace for blind people

Thomas Kieninger

November 1996 **Proceedings of the 9th annual ACM symposium on User interface software and technology**

Full text available: pdf(840.13 KB) Additional Information: full citation, references, index terms

Keywords: HTML editor, World Wide Web, blindness, document analysis, hypertext, pattern matcher

39 Toward a Dexter-based model for open hypermedia: unifying embedded references and link objects

Kaj Grønbæk, Randall H. Trigg

March 1996 **Proceedings of the the seventh ACM conference on Hypertext**

Full text available: pdf(1.31 MB) Additional Information: full citation, references, citings, index terms

Keywords: Dexter hypertext reference model, dynamic hypermedia, embedded links, generic links, link objects, open hypermedia

40 BIIF, VRML and CGM

Bernadette Kuzma, George S. Carson, Richard F. Puk, John Gebhardt

May 1999 **ACM SIGGRAPH Computer Graphics**, Volume 33 Issue 2

Full text available: pdf(1.58 MB) Additional Information: full citation, abstract, index terms

This edition of the Standards Pipeline consists of three articles that provide an in-depth look at applications of three important international standards. In the first article, Bernadette Kuzma of SEMCOR, Inc. describes the Basic Image Interchange Format (BIIF) formally known as ISO/IEC 12087-5:1998. While BIIF originated in the defense community, it is beginning to find wider applications in areas ranging from electronic libraries to medical imaging. In the second article, Richard F. Puk of In ...

The ACM Portal is published by the Association for Computing Machinery. Copyright ?2004 ACM, Inc.
Terms of Usage Privacy Policy Code of Ethics Contact Us

Useful downloads: Adobe Acrobat QuickTime Windows Media Player Real Player

This is Google's cache of http://www.grandcentral.com/developers/gettingstarted/architect/provision/ as retriev
Jul 25, 2004 19:03:05 GMT.
Google's cache is the snapshot that we took of the page as we crawled the web.
The page may have changed since that time. Click here for the current page without highlighting.
This cached page may reference images which are no longer available. Click here for the cached text only.
To link to or bookmark this page, use the following url: http://www.google.com/search?
q=cache:PcI4jAJDWFAJ:www.grandcentral.com/developers/gettingstarted/architect/provision/+provision+web+servic

Google is not affiliated with the authors of this page nor responsible for its content.

These search terms have been highlighted: **provision web service**

Developers

Provision New Connections Quickly and Easily

Downloads

.: **Provision** New Connections Quickly and Easily

To date, **service** oriented architectures have been most successful when they are deployed behind a firewall. That is because all the end points are known and are typically controlled by a common entity which can dictate the software used by all involved. However, deploying a **service** oriented architecture outside the firewall often proves to be much more difficult. That is because once you venture outside the firewall to connect to a third party you often lose the convenience of being able to dictate to them what software to deploy on their end in order to make the connection possible.

Grand Central provides a loosely coupled **Web** Services Network which allows you to isolate the complexity of your application to the network, eliminating the need for you to build out the same technology stack at each end point as you might have to with other B2Bi solutions. A loosely coupled architecture opens up new opportunities and possibilities because your partners no longer need to be limited by their technological capabilities. Deploying services on Grand Central not only lowers your cost per connection, but lowers the on-boarding and on-going management costs for your customers as well.

Furthermore, because of Grand Central's loosely coupled nature, your customers and partners are free to choose how they wish to connect to the Grand Central **Web** Services Network independently of how your application connects to the network. This gives you the immediate advantage of flexibility with your customers making it easier for them to connect to you as well, as well as cheaper, all the while being complete transparent to those you wish to connect to.

Did You Know?
Grand Central offers a suite of **Web** services that allow companies to greatly reduce the time spent by their customers in signing up on Grand Central, registering **Web** services, and provisioning other services to use them.

Developers:
Connect Your Application Now!

Additional Topics for Architects
Guarantee Corporate Security Policies Are Met
Acquire a Robust Infrastructure for Business Processes
Monitor **Service** Activity and Ensure Quality of **Service**
Facilitate Cross-Enterprise Integration
Extend Security Boundaries Beyond the Firewall
Provision New Connections Quickly and Easily

.: Develop
.: Quick S
.: Blueprir
.: Code Lil
.: TNT
.: Docume
.: Forums

Test Drive

Sign
25

Register N

Join
F

Sign Up for

enter email h

GRAND CENTRAL COMMUNICATIONS and the aardvark design are trademarks of
Grand Central Communications, Inc. in the United States and other countries.
All other trademarks are the property of their respective holders

Copyright © 2003-2004 Grand Central Communications, Inc. All rights reserved.

Mobile Networks and Applications
The Journal of SPECIAL ISSUES on Mobility of Systems, Users, Data and Computing

Article Abstract

Special Issue: Mobile and Wireless Data Management with Featured Section on Wireless Sensor Networks beginning on p. 425

Export Citation: Text RIS
doi:10.1023/A:1024583613701
Mobile Networks and Applications
8 (4): 389-399, August 2003
Copyright © 2003 Kluwer Academic Publishers
All rights reserved

Proxies + Path Prediction: Improving Web Service Provision in Wireless–Mobile Communications

Stathes Hadjiefthymiades
Communication Networks Laboratory, Department of Informatics and Telecommunications, University of Athens, Panepistimioupolis, Athens 15784, Greece

Lazaros Merakos
Communication Networks Laboratory, Department of Informatics and Telecommunications, University of Athens, Panepistimioupolis, Athens 15784, Greece

Abstract
Mobile computing is considered of major importance to the computing industry for the forthcoming years due to the progress in the wireless communications area. A proxy-based architecture for accelerating Web browsing in wireless customer premises networks is presented. Proxy caches, maintained in base stations, are constantly relocated to follow the roaming user. A cache management scheme is proposed, which involves the relocation of full caches to the most probable cells but also percentages of the caches to less likely neighbors. Relocation is performed according to the output of a user movement prediction algorithm based on a learning automaton. The simulation of the scheme shows considerable benefits for the end user.

Keywords
mobile computing, path prediction algorithm, caching proxy, cache relocation

Article ID: 5124151

©2004 Kluwer. All rights reserved.

This is Google's cache of http://msdn.microsoft.com/library/en-us/dnservice/html/service10172001.asp.
Google's cache is the snapshot that we took of the page as we crawled the web.
The page may have changed since that time. Click here for the current page without highlighting.
To link to or bookmark this page, use the following url: http://www.google.com/search?q=cache:2ft_p41MZ5EJ:msdn.microsoft.com/library/en-us/dnservice/html/service10172001.asp+web+service+endpoint&hl=en&ie=UTF-8

Google is not affiliated with the authors of this page nor responsible for its content.

These search terms have been highlighted: **web service endpoint**

Web Service Description and Discovery Using UDDI, Part II

Scott Seely
Microsoft Corporation

October 17, 2001

Introduction

In our previous column, we had a visit from Karsten Januszewski of the Microsoft® UDDI team. Karsten gave an overview of what UDDI is, why it exists, and how to use it. In this article, we are going to show what we had to do to register the Cold Rooster Favorites **Service** with the Microsoft UDDI registry. Because we had never registered with UDDI before, we had to start at the very beginning.

Registering Yourself with UDDI

We had never registered Cold Rooster with the Microsoft UDDI site, so our first order of business was to create an account on that site. Account registration requires you to log in using Passport. Once you log in, you have the option of setting the UDDI contact e-mail address to the one tied to your Passport account, or to something else. I opted to tie this to the crooster@microsoft.com address, since I would not be the only person utilizing UDDI on my team. We had already created this e-mail account for some other e-mail needs for Cold Rooster Consulting, so this seemed the logical choice.

The registration screen also asks for other contact information, such as the name of the registrant, a contact phone number, and a mailing address. To complete registration, you need to agree to a Terms of Use agreement (see Terms of Use).

Once you have your contact information in the registry and have agreed to the Terms of Use, the UDDI site will send you an e-mail so that the contact address can be confirmed. Click the link in that message, and you will be able to administer your UDDI account.

Administration involves adding business data to the registry, publishing tModels, and editing registration details. Figure 1 shows the administration page.

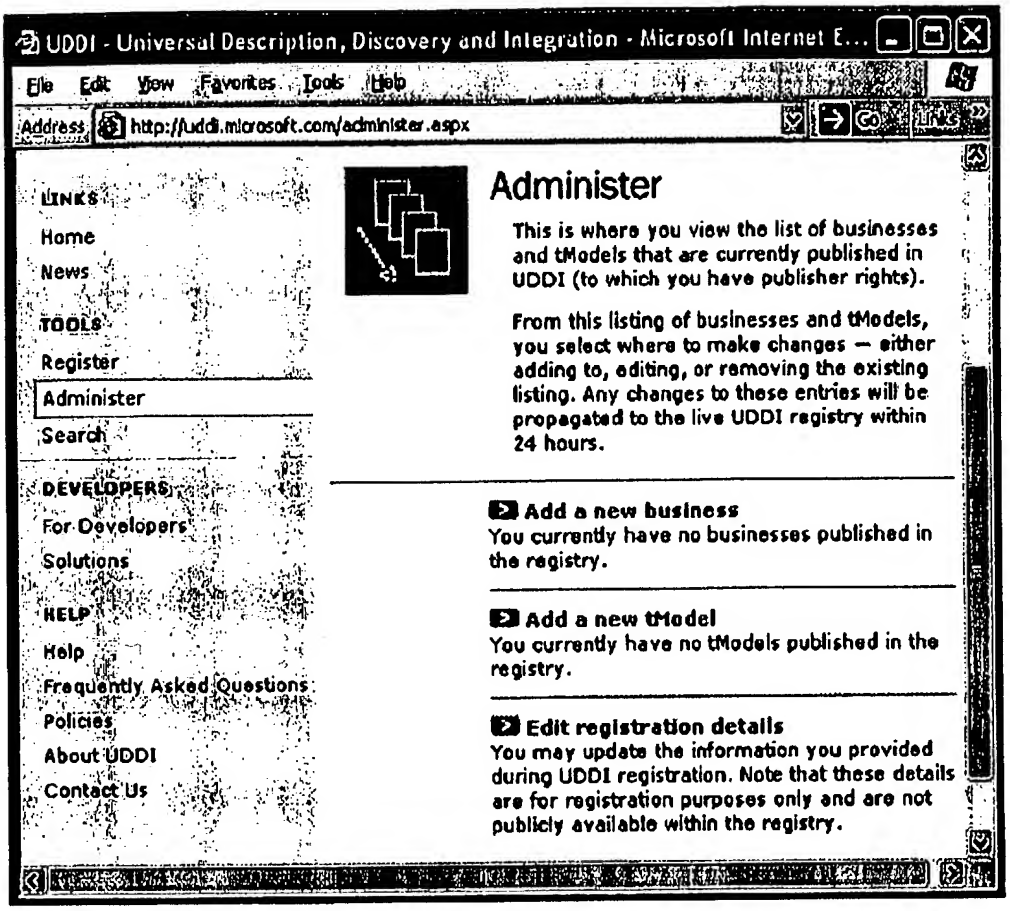


Figure 1. UDDI administration page

Add Your Business to UDDI

After establishing our account on the Microsoft UDDI site, the next task was to add Cold Rooster Consulting to the UDDI registry. We could register our business through the **Web** page or through the UDDI API using the .NET SDK or the COM SDK. I chose to use the UI, because I intended to register everything only once. If we someday got into the habit of changing bits of the interface on a regular basis, we would automate this process to reduce the chance for errors. Adding the business to the registry enables users to find us based on what we do and the types of **Web** Services we provide. To add Cold Rooster Consulting to the UDDI registry, we went to the "Add a new business" link on the administration page (Figure 1). The first page asks for the Business name and description. For Cold Rooster, I entered:

Name: Cold Rooster Consulting

Description: Fictional company used by MSDN Architectural Samples Team

With the business added to the UDDI registry, I could now add more information:

Best Available Copy

- **Contacts:** People to contact who can help out with anything provided by the business. We added various members of the MSDN Architectural Samples team to the list of contacts.
- **Services:** The tModels (WSDL files) that this company exposes. We added the Account, Logon, and Report tModels to our UDDI entry.
- **Identifiers:** Pieces of data unique to this business, such as a company register listing number. Since Cold Rooster does not have one of these, we did not fill this out.
- **Business classifications:** These identify the location of the business, as well as what the business does. Cold Rooster Consulting is located in the state of Washington within the United States.
- **Discovery URLs:** This provides a location where details about the business can be found.

We will do the simpler items first and save the services for last.

Adding Contacts to UDDI

This is pretty simple. Like any other address form, you just fill in the general details for the various contacts. Based on the description and usage notes, potential users refer to this information to contact your company to license the **Web Service**, obtain support, or contact you about other business-related items. Figure 2 shows how the form looks for my contact information.

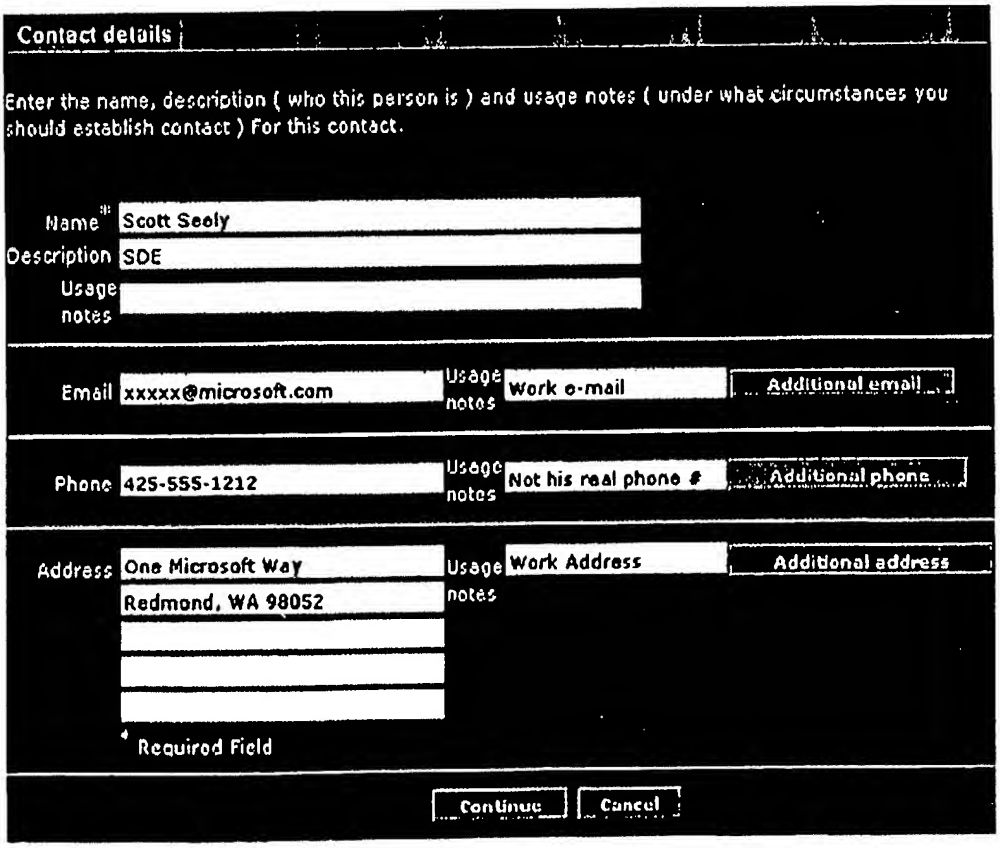


Figure 2. Contact details page

Best Available Copy

Classifying the Business

Most businesses can be classified by what they do. When classifying entities, a number of taxonomies are available for UDDI classification. You can use the following:

- North American Industry Classification System (NAICS-1997)
- Universal Standard Products and Services Codes (UNSPSC-7.03)
- ISO 3166 Geographic Taxonomy
- Standard Industrial Classification (SIC-1987)
- GeoWeb Geographic Classification
- UDDI Types Taxonomy
- We registered Cold Rooster under all of these taxonomies except for UDDI types. We did not register the business under the UDDI types taxonomy, because that particular taxonomy exists to classify tModel and **service** information.
- To know what we want to register, we have to look at what Cold Rooster does and where it is located. Cold Rooster Consulting is based in Redmond, Washington, USA. It provides project-based and supplemental staffing computer consulting services. Furthermore, it specializes in Windows and Internet-based development. Knowing all of this, we needed to appropriately classify the business in each of the six classification schemes. Table 1 shows how we classified the business for each scheme.

Classification Scheme	Classification
NAICS	• 541511: Custom Computer Programming Services • 541512: Computer Systems Design Services
UNSPSC	• 81.11.16.07.00: Programming for C or C++ • 81.11.16.03.00: Programming for HTML • 81.11.16.01.00: Programming for Microsoft® Visual Basic® • 81.11.16.12.00: Programming or Proprietary Languages (we do C# too) • 81.11.21.06.00: Application **Service** Providers (we host **Web** Services) • 81.11.21.03.00: World Wide **Web** (WWW) site design services
ISO 3166	• US-WA (Washington, USA, World)
SIC	• 7371: Computer programming services • 7372: Information retrieval services
GeoWeb Geographic Classification	• 518816 (Redmond, Washington, USA, North America, World)

Table 1. Sample UDDI classifications

Our business is now fully classified. The next step is to add the three tModels.

Adding the tModels

Just in case you missed last week's article, a tModel is a type model. For **Web** Services, tModels are typically synonymous with WSDL files. As such, they define the types used by the **Web Service**, as well as the message and operation definitions for the **Web Service**. Given a tModel, I know what **Web Service** operations are implemented by an entity implementing that tModel, and how to access those operations. You register your WSDL files as tModels because there may eventually be multiple implementations of these tModels.

The Server Side Favorites **Web Service** is composed of three **Web** Services: Logon, Account, and Report. The Logon **Web Service** allows a licensee to login and get a token. Using that token, the licensee can access the other methods in the Account and Report **Web** Services. To add the WSDL files, you need them deployed on a server that is reachable over the public Internet. This is also where you use the UDDI classification.

To add a tModel, just click on "Add a new tModel" from the administration page (Figure 1). You then need to add some basic information for the tModel: its name, a description, and where to locate the WSDL document. Once you do that, you then classify the tModel as a WSDL document and publish the information to the UDDI registry. It's all pretty simple. Figure 3 shows the details filled in for the Logon tModel.

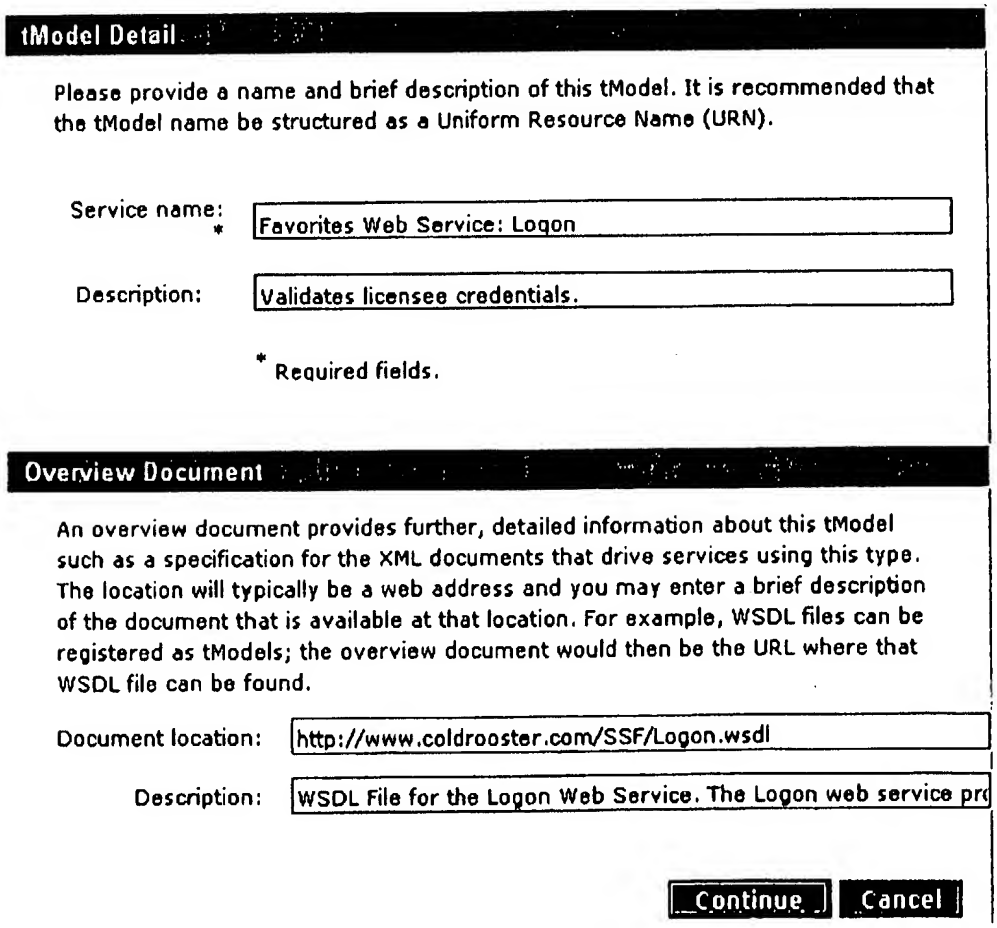


Figure 3. Logon tModel details

On the next screen, you can add **service** classifications and business identifiers. This is

the same data you could enter for the business information. As a **service** classification, we only used the UDDI classification. You will click through a series of links to specify this information:

- These types are used for tModels
- Specification for a **Web Service**
- Specification for a **Web Service** described in WSDL
- In registering Cold Rooster Favorites **Service**, I followed these steps for Logon.WSDL Account.WSDL, and Report.WSDL. The UDDI registration was then completed and the information available in the registry.

Defining the Services

Once the tModels are in place, you still have to add the services to declare that these services exist. To do this, go back to the administration page and, this time, you will see your business listed under "Add a new business" (Figure 1). Select your business and scrol down to "Services." Here, click "Add a **Service**." On the first page you will fill in the **Service** detail. For the Logon **Web Service**, I filled in the following:

- Name: Logon
- Description: Validates licensee and provides an access token

Once that was done, I classified the **Service** once again as a "Specification for a **Web Service** described in WSDL." I then bound the **Service** to the Logon tModel that was already registered. In "Define a new binding," I filled in the fields as shown below:

- Access point: https://Coldrooster.com/SSF/Logon.asp
- URL type: http
- Description: Cold Rooster Consulting Logon **Web Service endpoint**

To finish this, I had to associate the **Service** with the Favorites **Web Service**: Logon tModel. Under Specification Signatures, select "Add specification signature." To look the model up by name, enter "Favorites **Web Service**." This will bring up all three of the tModels saved for the Favorites **Service**. Select "Favorites **Web Service**: Logon" and press "Continue." A **Web** page then appears on which you need to edit some more information about the **endpoint**. I filled in those fields as follows:

- Edit specification signature; Description: Cold Rooster implementation of the Favorite **Web Service**: Logon tModel
- Instance details; Parameters: http://msdn.microsoft.com/library/?url=/library/en-us/dncold/html/ssfapiref.asp?frame=true
- Instance details; Description: API Reference document
- Overview document; Document location: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dncold/html/ssf1sec.asp
- Overview document; Description: Server Side Favorites Security information

With that filled in, I clicked "Continue" once more. I could then review all the information

entered about the Logon **service** before returning to the main Business view of the data and publishing everything to the UDDI registry. I then repeated this process for both the Account and Report **Web** Services.

Finding the Data

Today (October 8, 2001) only one company is listed in Redmond according to the GeoWeb Taxonomy: Cold Rooster Consulting. I did another search, this time by ISO 3166 classification. This time, nine companies appeared. Other classifications brought up other totals. For the other classification schemes, I typically wound up with several pages of results.

Because the data is now discoverable, people interested in **Web** Services that store favorite URLs will theoretically flock to the Cold Rooster solution to use it. For those using Microsoft® Visual Studio® .NET, using UDDI to find the **Web Service** and add it to their project will be simple.

When you first go to the Add **Web** Reference dialog (ProjectAdd **Web** Reference), you are presented with a dialog that allows you to look for a **Web** Reference (also known as a WSDL file) through the Microsoft UDDI server. The **endpoint** used for Visual Studio is http://uddi.microsoft.com/visualstudio/. When I tell that **endpoint** to search for all businesses starting with "cold," it only finds the three **Web** Services I registered: Account, Logon, and Report. To add a **Web** Reference to the Logon **Web Service**, I just click "Logon" to expand the node, and then "Favorites **Web Service**: Logon" to view the WSDL At this point, I click "Add Reference" and I am ready to Logon to the Favorites **Web Service**.

If you have been following along up to this point, but do not have a Favorites license, just go to the Favorites **Service** Admin Console and sign up for one. You should get your password sent back to you within 15-30 minutes. To connect to the Logon **Web Service** using Visual Basic, the code is this simple:

```
Sub Main()
    Dim svc As New com.coldrooster.www.Logon()
    System.Console.WriteLine(svc.Logon("LicenseeName", "Password"))
    System.Console.WriteLine("Press return to exit")
    System.Console.ReadLine()
    svc.Dispose()
End Sub
```

This code prints out the GUID token used to gain access to the other methods available to the Favorites **Web Service**. The code for any of the other methods will be equally easy to write.

Summary

By registering your business, custom tModels, and **Web** Services with UDDI, you help developers all over the world to find your **Web** Services. The UDDI registry allows you to expose much more than **Web Service** endpoints and business data. The people using

UDDI can also use the interface to find **Web Service** documentation and samples. Usage of the Microsoft UDDI registry is free. You are encouraged to register your **Web** Services and business in this registry. Please take some time and familiarize yourself with http://uddi.microsoft.com. You might try to find the Cold Rooster Consulting information and browse it to see all the information that is available. Finally, now would be a great time to use Visual Studio .NET to attach to the Favorites **Service** and do some experimenting of your own.

Next time, we will have a guest column from Allen Wagner. Allen will be discussing techniques for handling large SOAP messages.

At Your Service

Scott Seely is a member of the MSDN Architectural Samples team. Besides his work there, he has published two books through Prentice Hall: *SOAP: Cross Platform **Web Service** Development Using XML* and *Windows Shell Programming.* He wrote and maintains a smal C++ based SOAP library (http://www.scottseely.com/soap.htm), published under the LGPL.

Contact Us | E-Mail this Page | MSDN Flash Newsletter

© 2003 Microsoft Corporation. All rights reserved. Terms of Use Privacy Statement Accessib